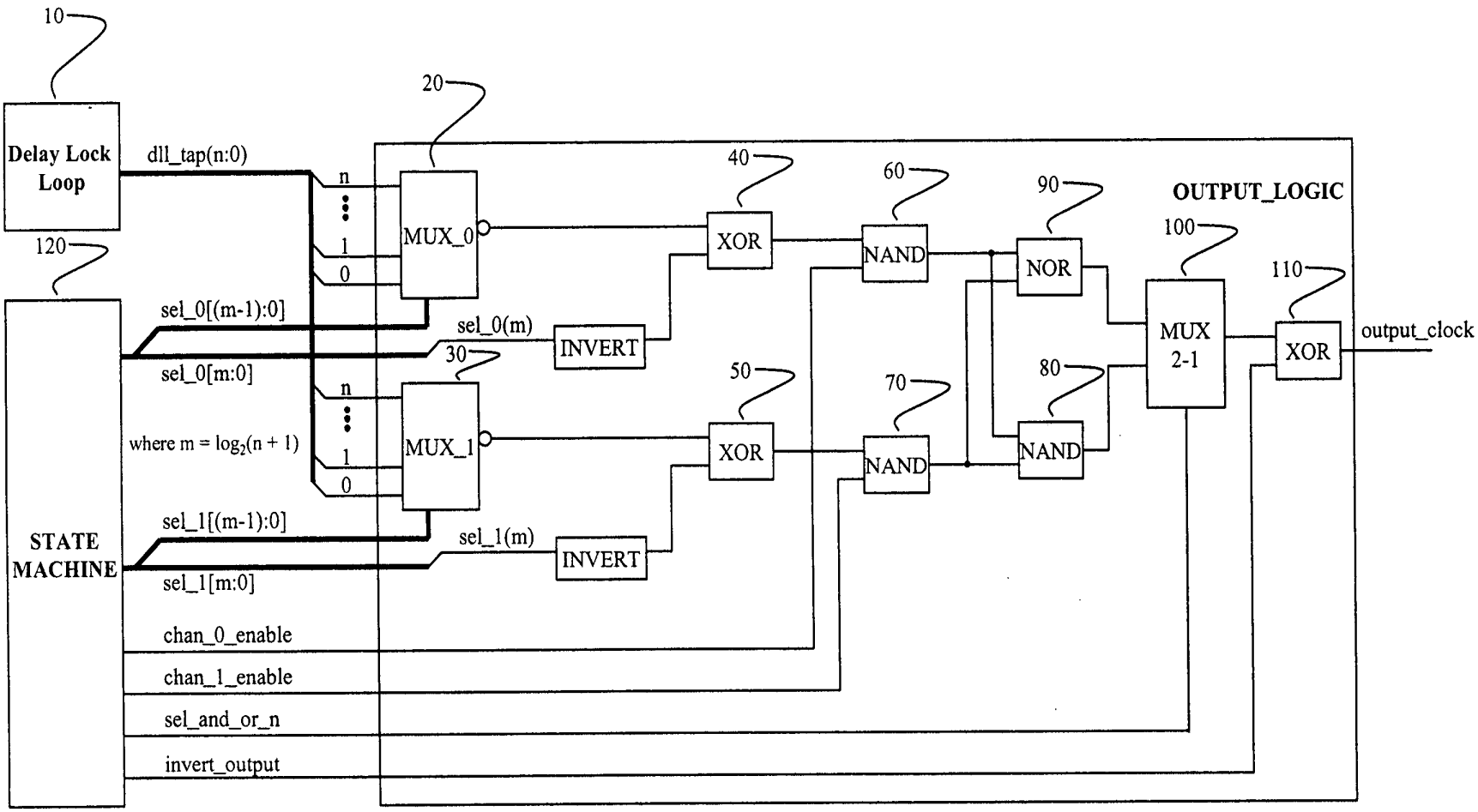
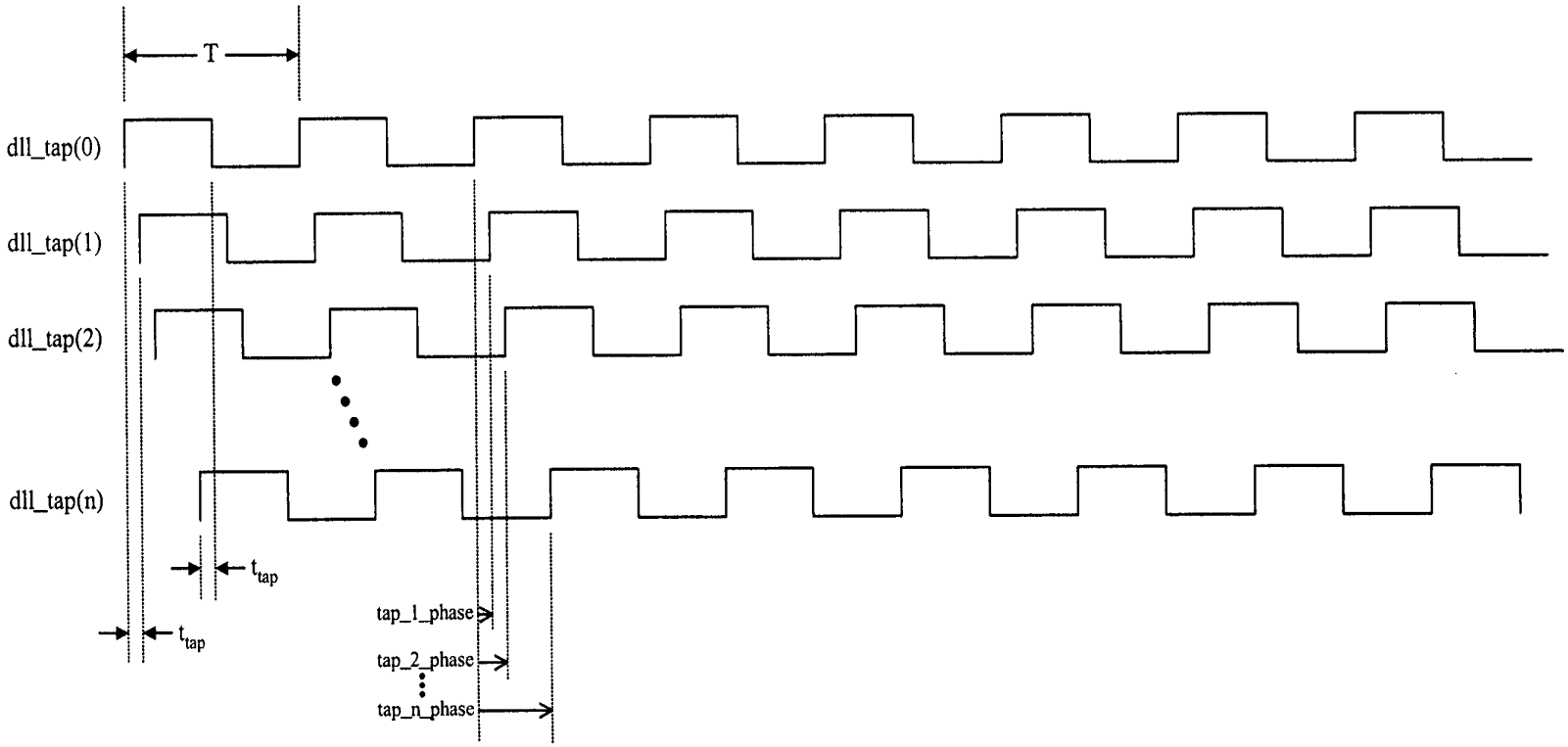

10
Delay Lock Loop
dll_tap(n:0)
20
MUX_0
n
1
0
40
XOR
60
NAND
90
NOR
OUTPUT_LOGIC
100
MUX 2-1
110
XOR
output_clock
120
sel_0[(m-1):0]
sel_0(m)
INVERT
sel_0[m:0]
30
MUX_1
50
XOR
70
NAND
80
NAND
where m = log2(n + 1)
sel_1[(m-1):0]
sel_1(m)
INVERT
STATE MACHINE
sel_1[m:0]
chan_0_enable
chan_1_enable
sel_and_or_n
invert_output

Fig. 1



t_{tap} = (1 / [(n+1) * 2]) * T, where T = clock period, n = number of DLL taps = 1/2 the number of output clock offset/width positions within one period, T.

tap_0_phase = t_{tap} * 0 = 0

tap_1_phase = t_{tap} * 1

tap_2_phase = t_{tap} * 2

⋮

tap_n_phase = t_{tap} * n

Fig. 2

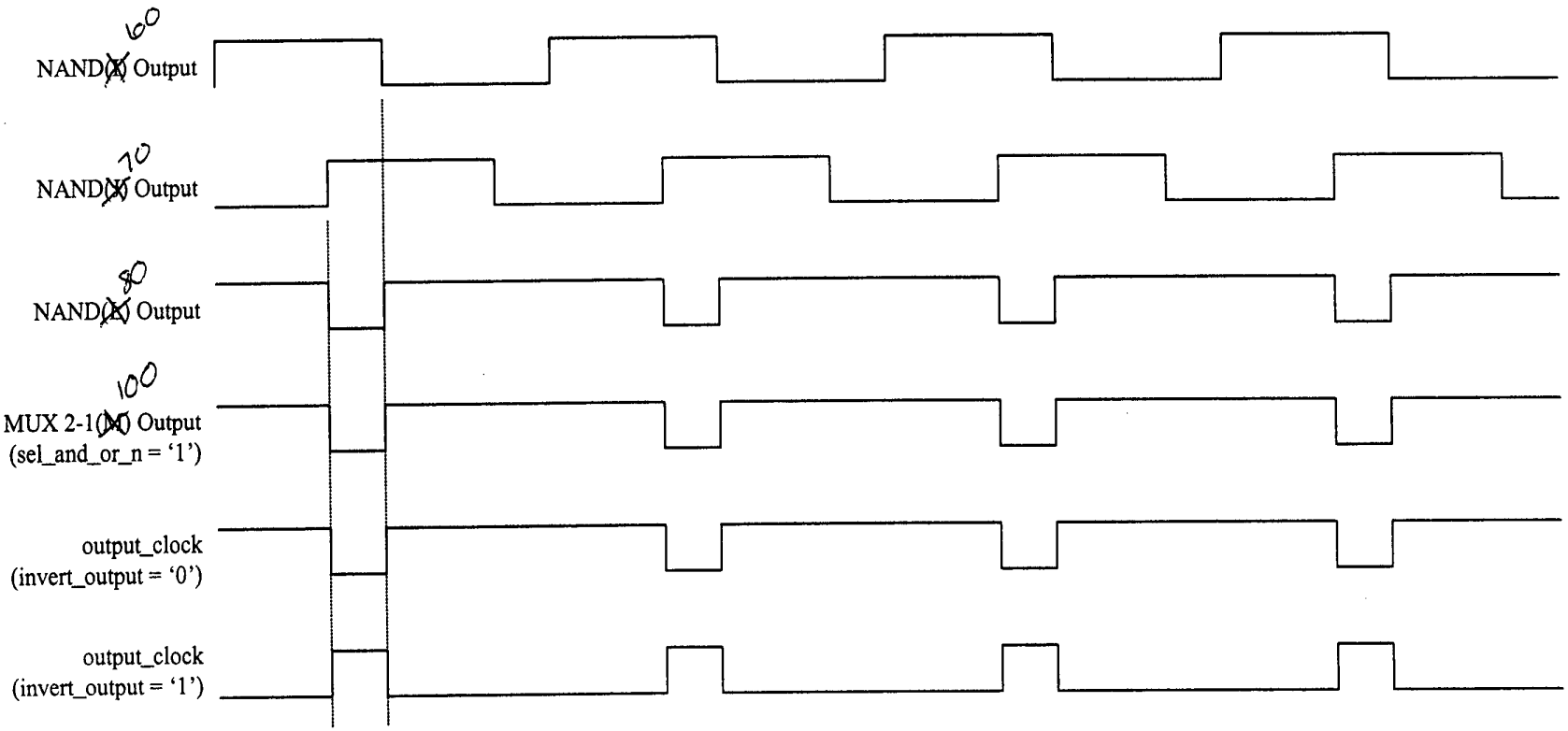

60
NAND(X) Output
70
NAND(X) Output
80
NAND(X) Output
100
MUX 2-1(X) Output
(sel_and_or_n = '1')
output_clock
(invert_output = '0')
output_clock
(invert_output = '1')

Fig. 3

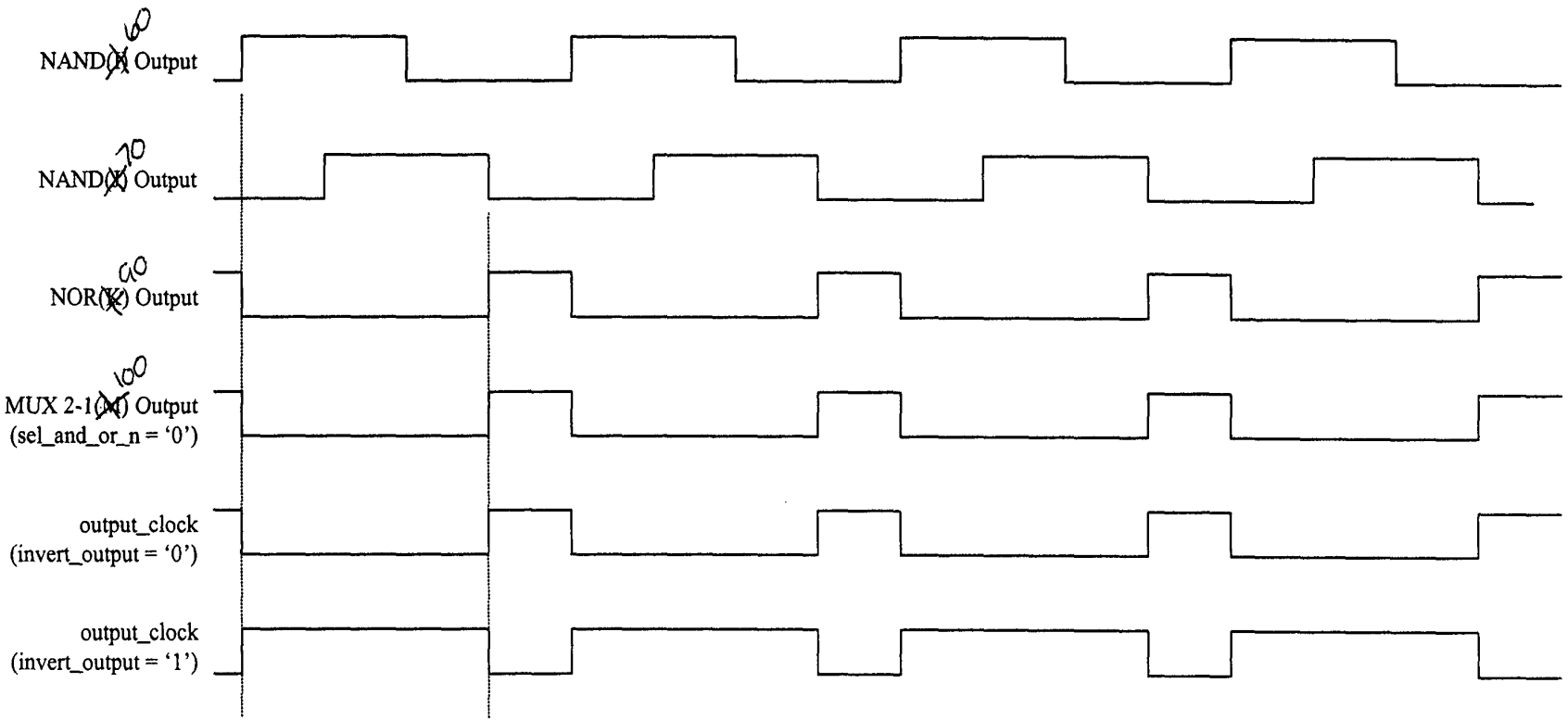

NAND(60) Output
NAND(70) Output
NOR(80) Output
MUX 2-1(100) Output
(sel_and_or_n = '0')
output_clock
(invert_output = '0')
output_clock
(invert_output = '1')

Fig. 4